Hierarchical Storage Resource Manager (HRM) for Grid Architecture

Alex Sim
Arie Shoshani

Scientific Data Management Group
Computing Science Directorate
Lawrence Berkeley National Laboratory

Globus Retreat 2000

Outline

- HRM role in the Data Grid architecture
- HRM API description
- HRM - HPSS system functionality
- File tracking

HRM role in the Data Grid architecture

- The class of Storage Resource Managers includes:
  - HRM: for managing the access to tape resources
    - may or may not have a disk cache
    - functionality generic but needs to be specialized for specific mass storage systems
    - e.g. HRM-HPSS, HRM-Enstore, ...
  - DRM: for managing disk resources
    - functionality generic but needs to be specialized for specific disk systems
    - e.g. DRM-FileSystem, DRM-DPSS, …

HRM role in the Data Grid architecture

Functionality Examples

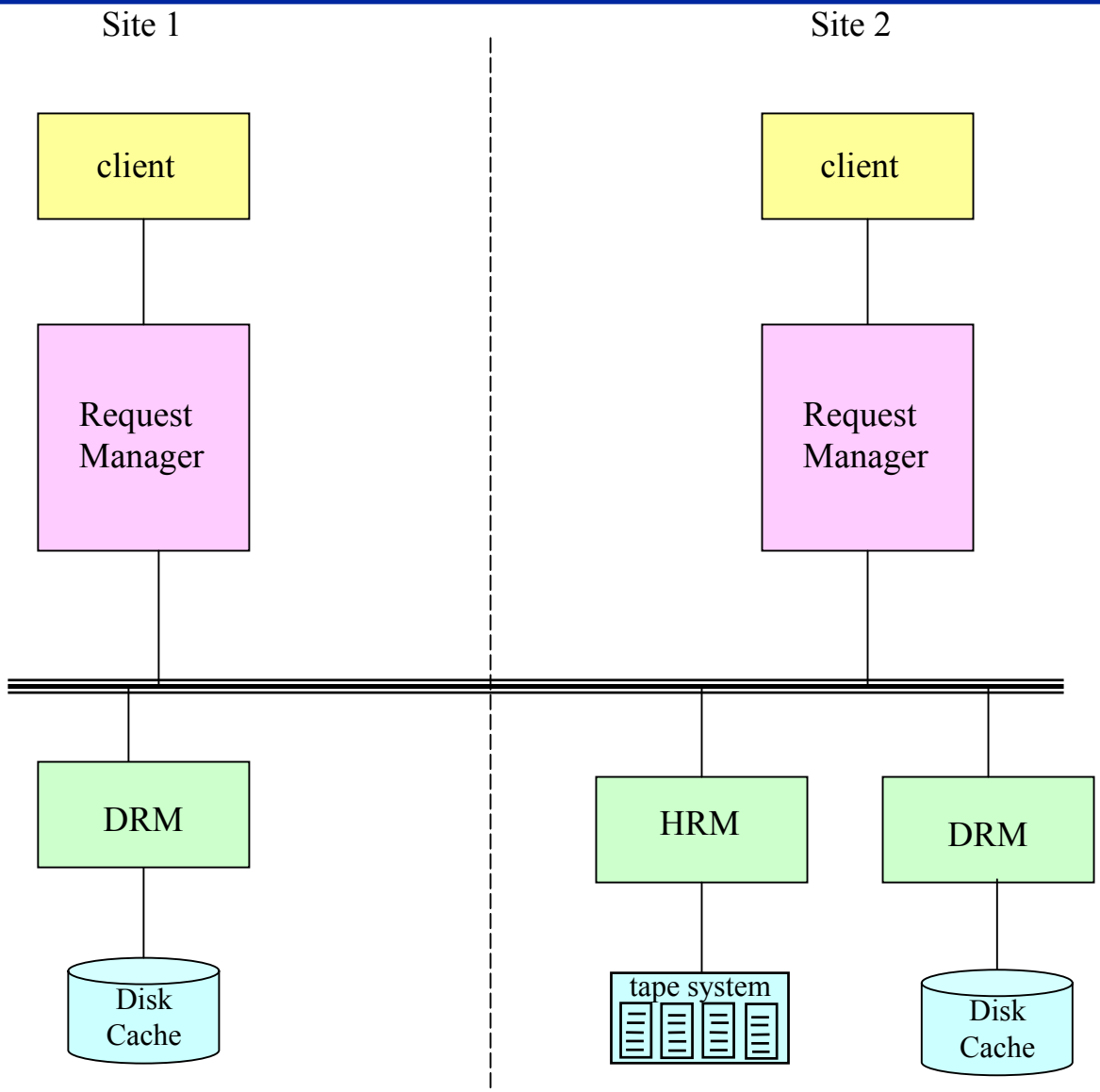
HRM functionality may include :

-- queuing of file transfer requests
-- reordering of request to optimize file transfer
(ordered by files on the same tape)
-- Monitoring progress and error messages
-- reschedules transfers that failed

DRM functionality may include:

-- keeping tracks of files in cache
-- making decisions on which files to remove when space is needed
-- optimizing cache use – by file sharing for multiple clients when requested files are already in cache

HRMs & DRMs in a Grid Architecture



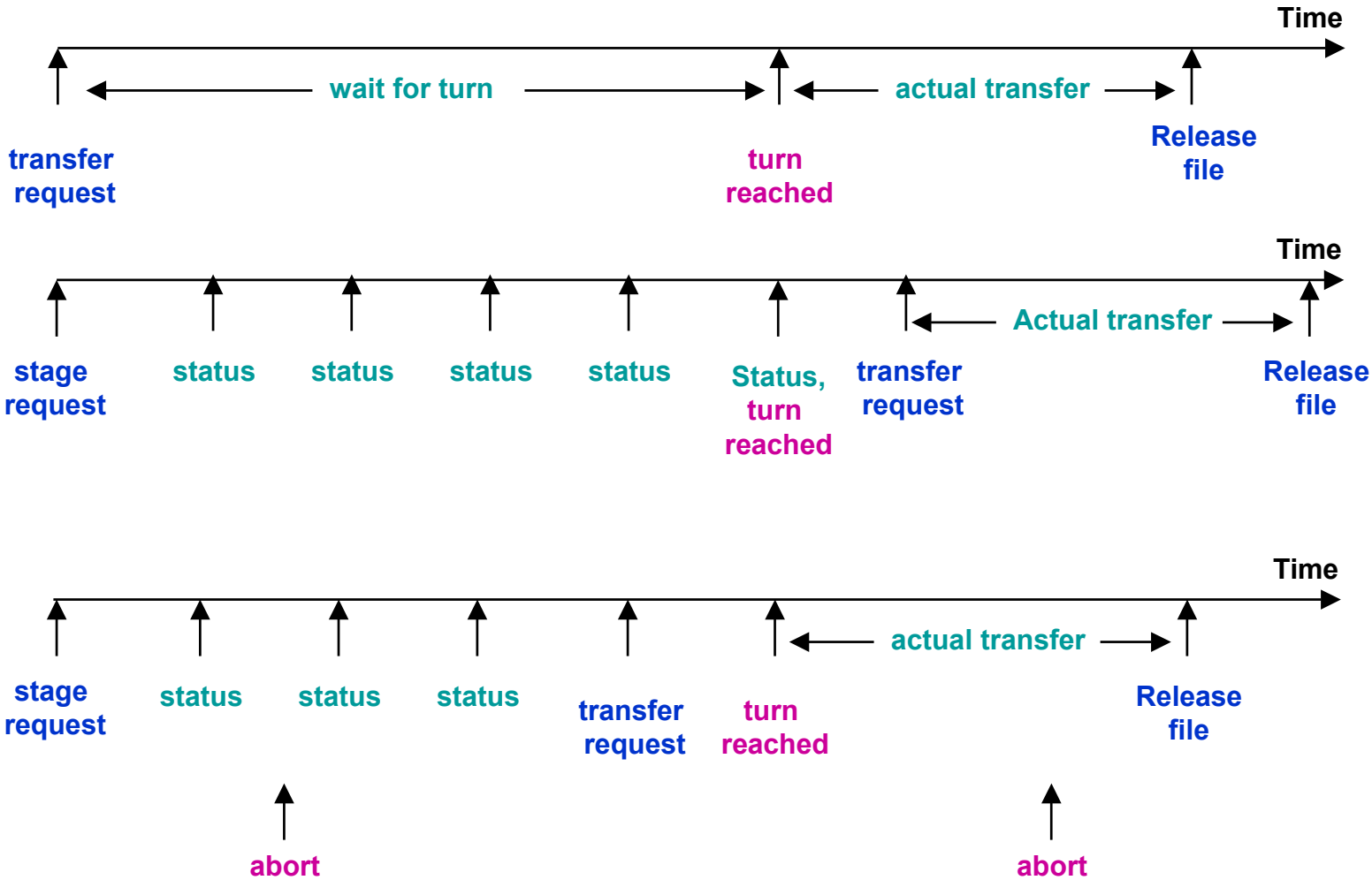
Advantages:

1. Each SRM has global view of all requests to it.
2. SRMs support reservation requests for space and file transfers
3. Can develop uniform error handling codes for all resources
4. Can ask SRMs to queue and monitor requests.
5. Main implication: need APIs to invoke SRMs, and a way for the Grid to pass requests to them.

HRM API description

- API Functionality
  - Request to transfer a file to destination disk
    - A blocking call
  - Request to stage a file to HRM disk
    - A non-blocking call when HRM disk exists
  - Request status/time_estimate
    - How long before file request will be processed
  - Request to abort a file transfer or stage
    - In case file no longer needed
  - Release a file
    - After file was moved to destination
    - optional to improve system efficiency
  - Call back when file is staged

Examples of call sequences



IDL for HRM


```
// hrm.idl
// IDL for HPSS Resource Manager
//
#ifndef HRM_IDL
#define HRM_IDL

enum hrmStatus {
        HRM_TRANSFER_COMPLETED,
        HRM_TRANSFER_FAILED,
        HRM_STAGE_ACCEPTED,
        HRM_STAGE_COMPLETED,
        HRM_STAGE_REFUSED,
        HRM_STAGE_FAILED,
        HRM_RELEASE_DONE,
        HRM_RELEASE_FAILED,
        HRM_ABORT_DONE,
        HRM_ABORT_FAILED,
        HRM_FILE_DOES_NOT_EXIST,
        HRM_HPSS_DOWN,
        HRM_HPSS_ERROR
};

module HPSSResourceManager {
        interface hrmGrid {
                hrmStatus transferFile
                        (in string sourceUrl,
                         in string destinationUrl,
                         in long waitTime,
                         in short userPriority,
                         out double timeEstimate);
                hrmStatus stageFile
                        (in string reference,
                         in string destinationUrl,
                         in long waitTime,
                         in short userPriority,
                         out double timeEstimate);
                boolean abort(in string destinationUrl);
                double timeEstimate(in string destinationUrl);
                boolean releaseFile(in string destinationUrl);
        };
};
#endif
```

HRM system functionality

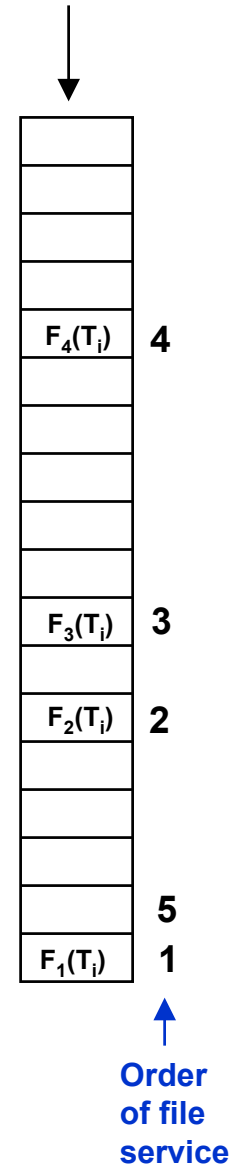
- Currently implemented for HPSS system
- All transfers go through HRM disk
  - reasons: flexibility of pre-staging
  - disk is sufficiently cheap for a large cache
  - opportunity to optimize for same file requests
- Functionality
  - queuing file transfers
  - file queue management
  - File clustering parameter
  - Transfer rate estimation
  - Query estimation - total time
  - Error handling

Queuing File Transfers

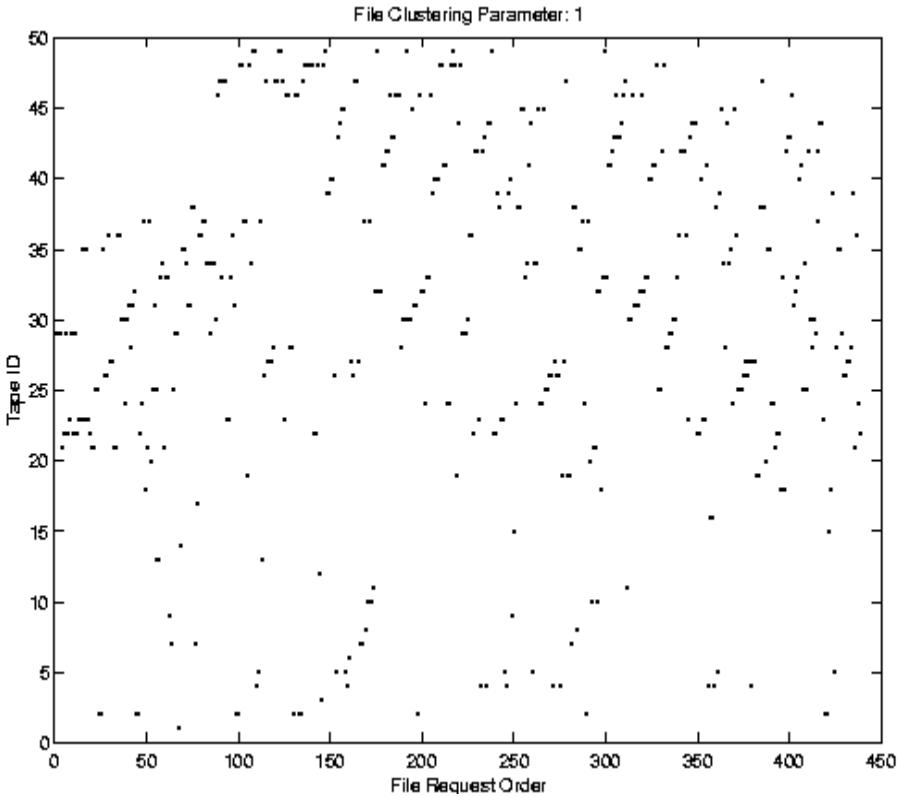
- Number of PFTPs to HPSS are limited
  - limit set by a parameter - NoPFTP
  - parameter can be changed dynamically
- HRM is multi-threaded
  - issues and monitors multiple PFTPs in parallel
- All requests beyond PFTP limit are queued
- File Catalog used to provide for each file
  - HPSS path/file_name
  - Disk cache path/file_name
  - File size
  - tape ID

File Queue Management

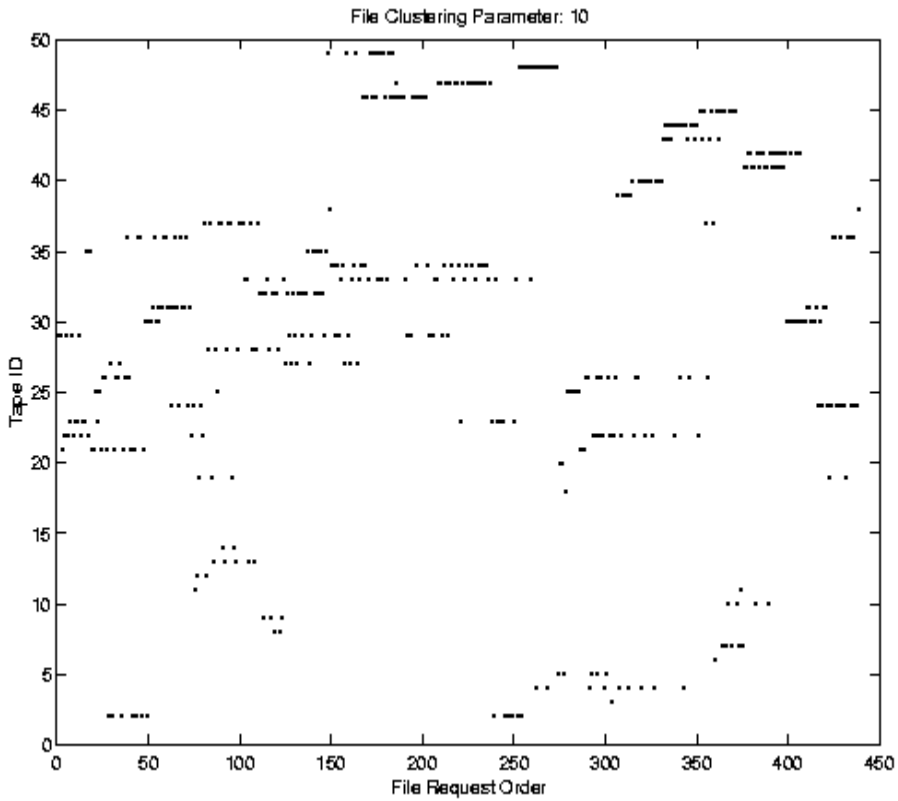
- Goal
  - minimize tape mounts
  - still respect the order of requests
  - do not postpone unpopular tapes forever
- File clustering parameter - FCP
  - If the file at top of queue is in $Tape_i$ and FCP > 1 (e.g. 4) then up to 4 files from $Tape_i$ will be selected to be transferred next
  - then, go back to file at top of queue
- Parameter can be set dynamically



File Caching Order for different File Clustering Parameters



File Clustering Parameter = 1



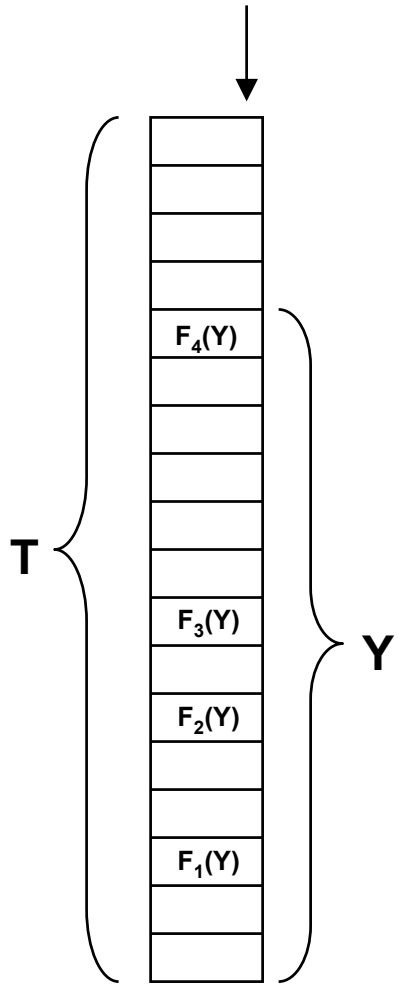
File Clustering Parameter = 10

Transfer Rate (Tr) Estimates

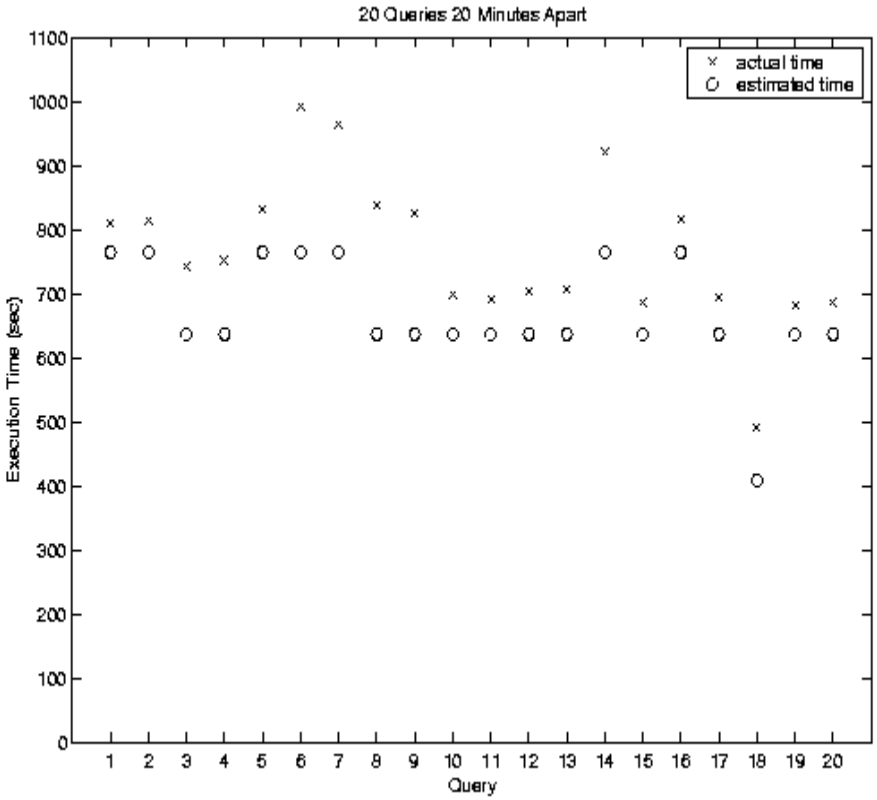
- Need Tr to estimate total time of a query
- Tr is average over recent file transfers from the time PFTP request is made to the time transfer completes. This includes:
  - mount time, seek time, read to HPSS Raid, transfer to local cache over network
- For dynamic network speed estimate
  - check total bytes for all file being transferred over small intervals (e.g. 15 sec)
  - calculate moving average over n intervals (e.g. 10 intervals)

Query Estimate

- Given: transfer rate Tr.
- Given a query for which:
  - X files are in cache
  - Y files are in the queue
  - Z files are not scheduled yet
- Let s(file_set) be the total byte size of all files in file_set
- If Z = 0, then
  - QuEst = s(Y)/Tr
- If Z ≠ 0, then
  - QuEst = (s(T)+ q.s(Z))/Tr
    where q is the number of active queries

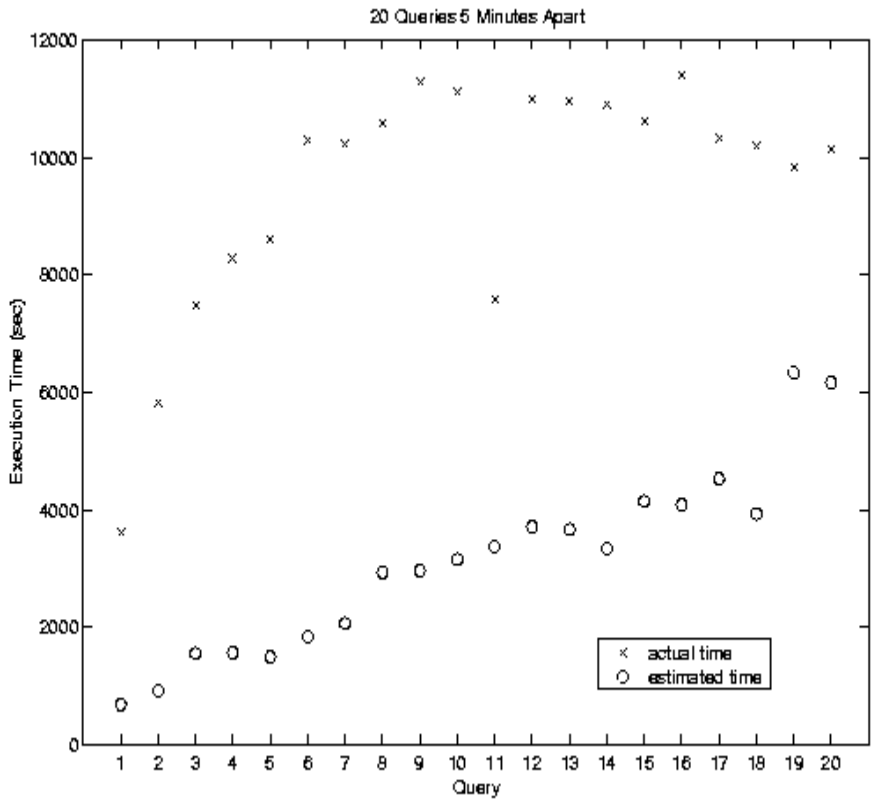


Reason for having q.s(Z)



20 Queries of length ~20 minutes launched 20 minutes apart

Estimate pretty close



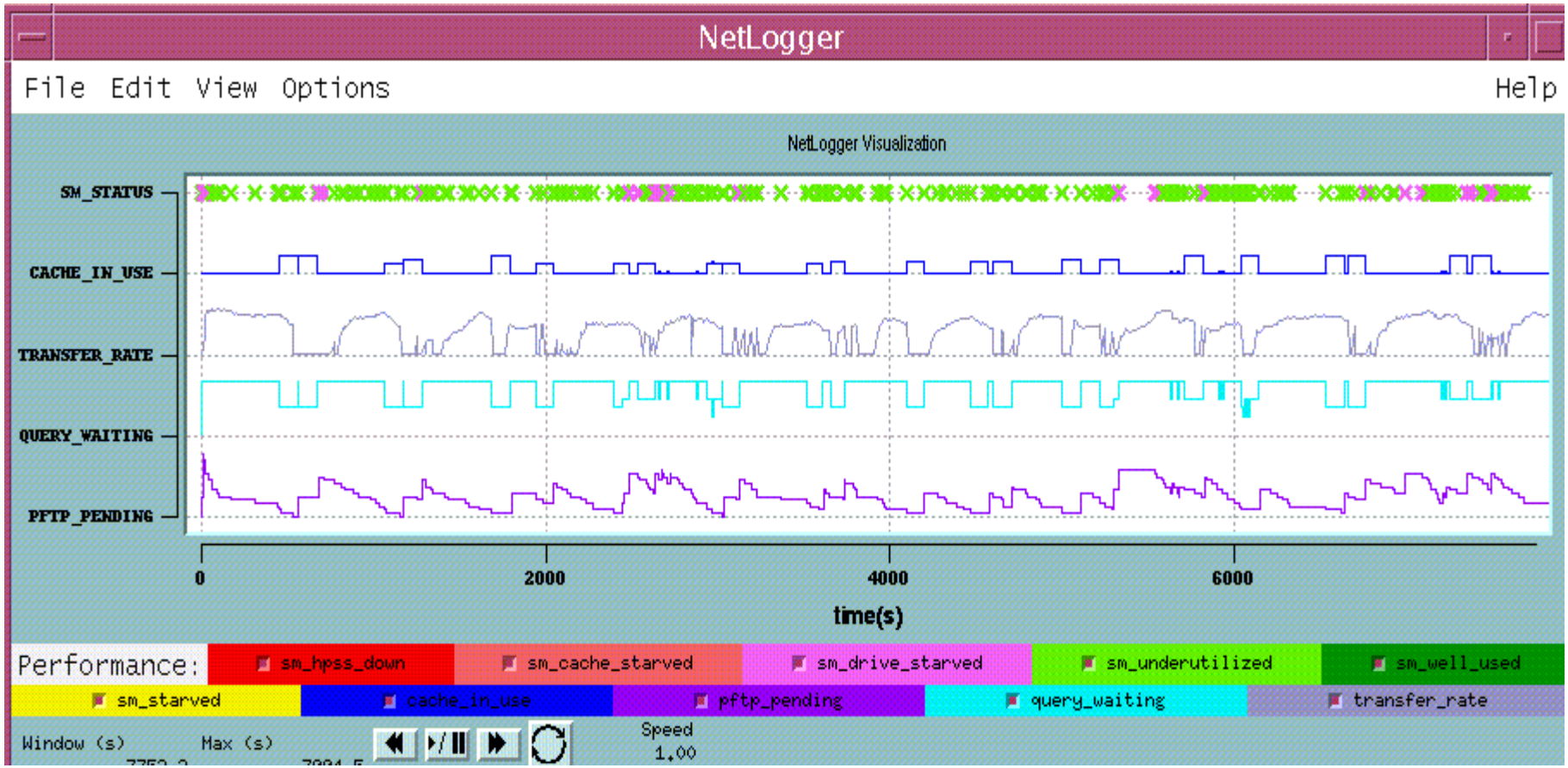
20 Queries of length ~20 minutes launched 5 minutes apart

Estimate bad - request accumulate in queue

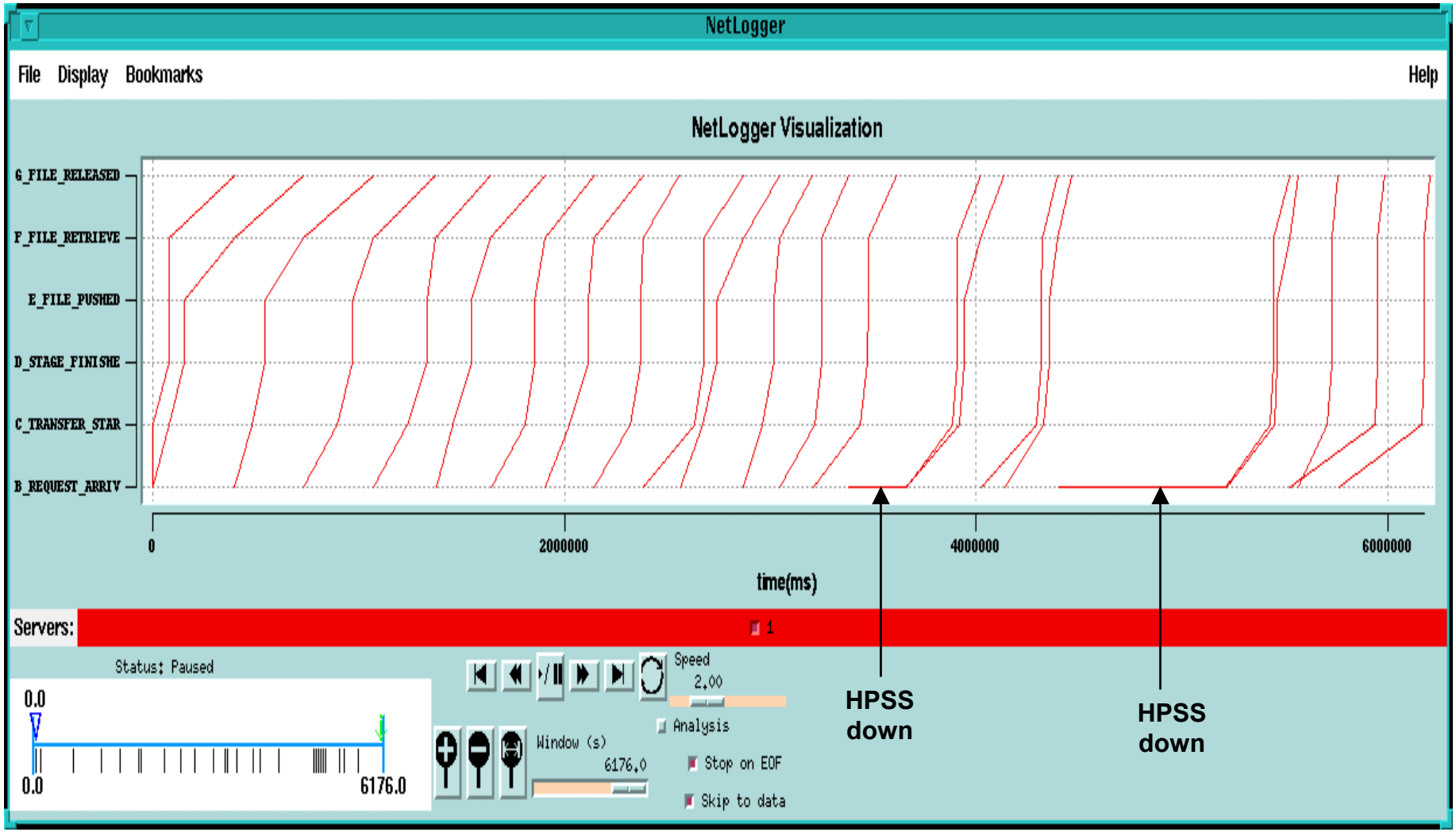
Error Handling

- 5 generic errors
  - file not found
    - return error to caller
  - limit PFTP reached
    - can't login
    - re-queue request, try later (1-2 min)
  - HPSS error (I/O, device busy)
    - remove part of file from cache, re-queue
    - try n times (e.g. 3), then return error "transfer_failed"
  - HPSS down
    - re-queue request, try repeatedly till successful
    - respond to File_status request with "HPSS_down"

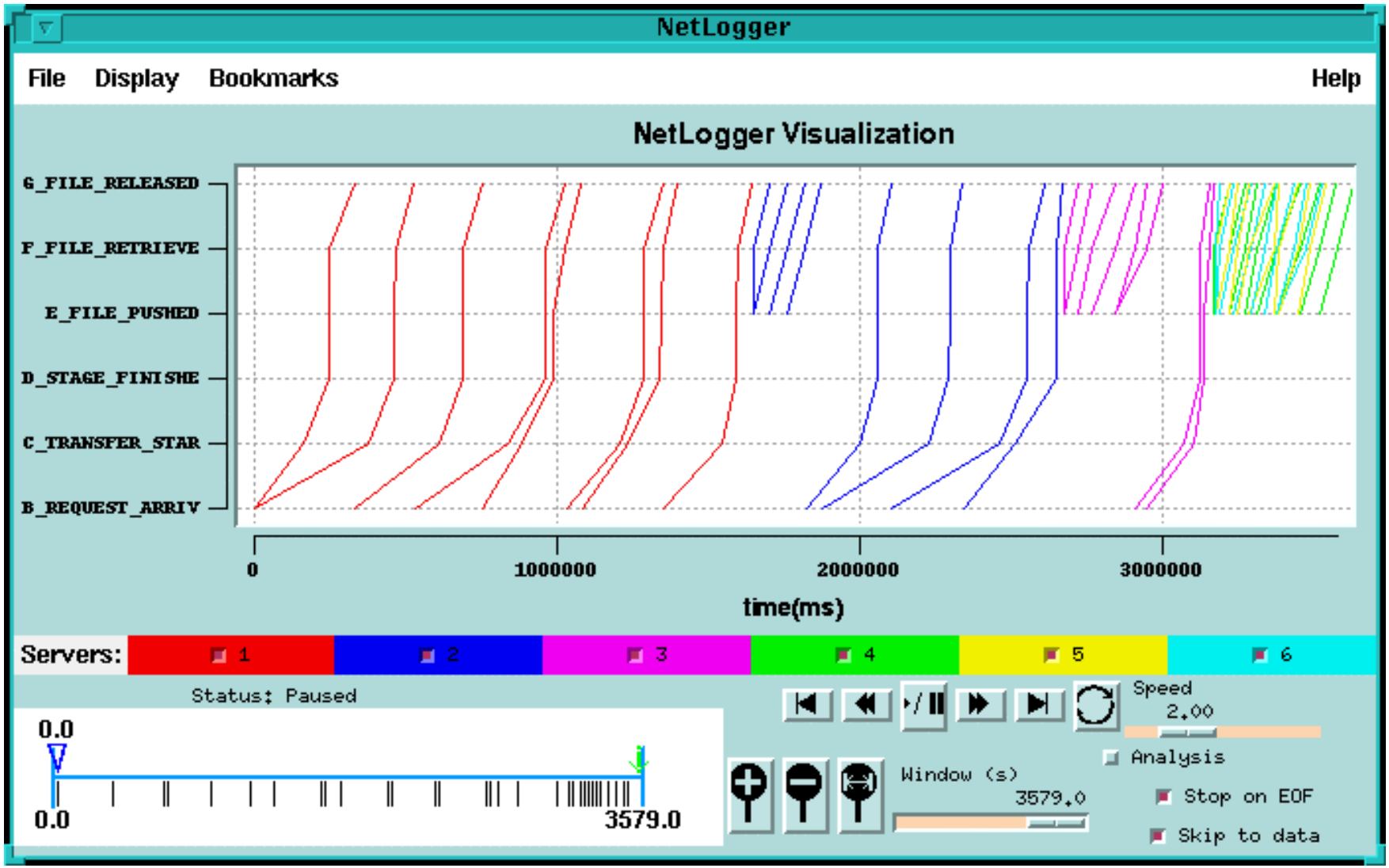
Dynamic Display of Various Measurements



File Tracking (1)



File Tracking (2)



Summary

- HRM performs tape resource management
  - queues transfer requests
  - monitors transfer progress
  - perform pre-staging
  - insulate requester from MSS failures
  - provides time estimates for file transfer
  - can support local system policy
    - number of simultaneous file transfer requests to MSS
    - number of file staging requests per user
    - support user priorities
    - etc.

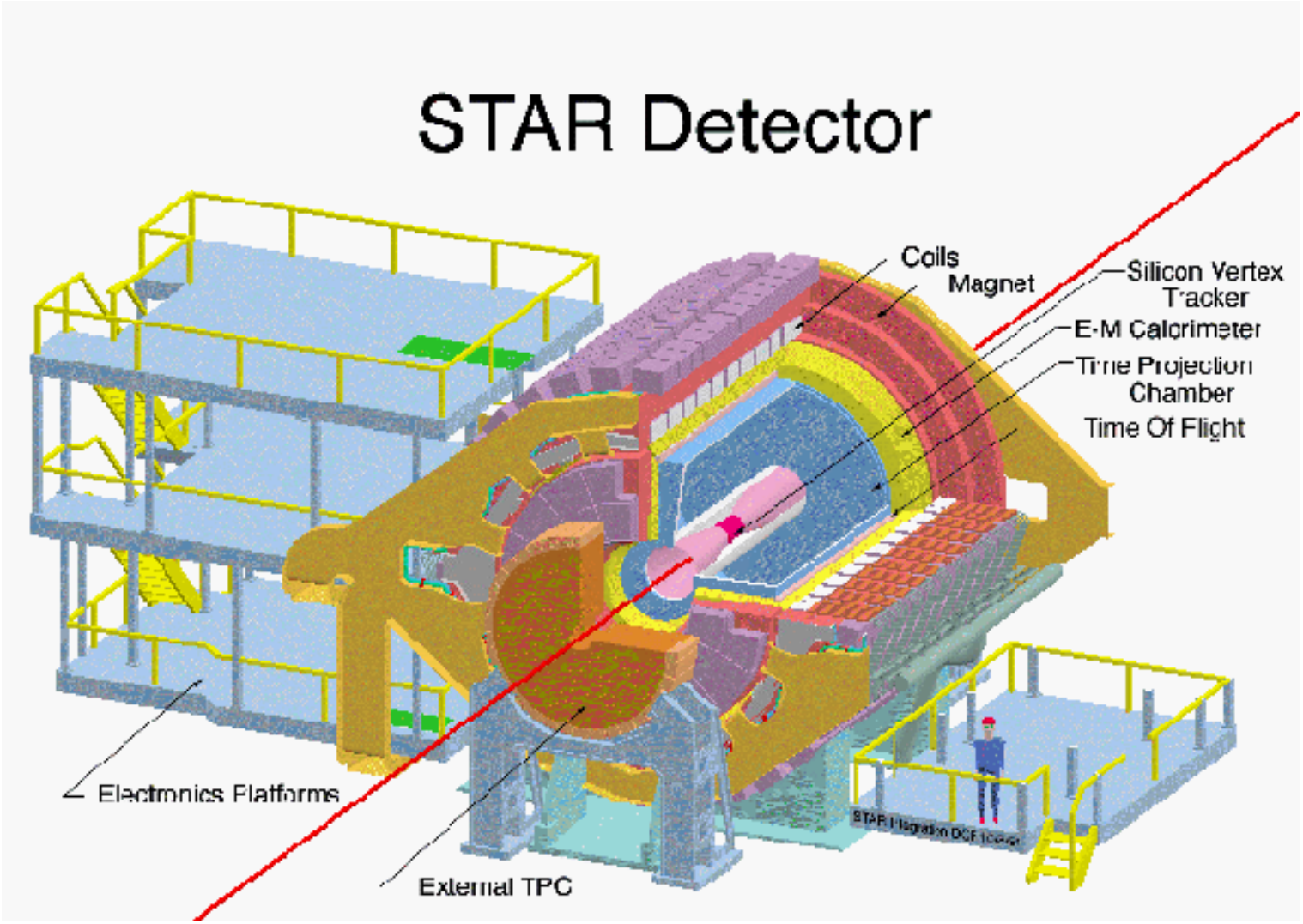
Optimizing Storage Management for High Energy Physics Applications

Data Volumes for planned HENP experiments

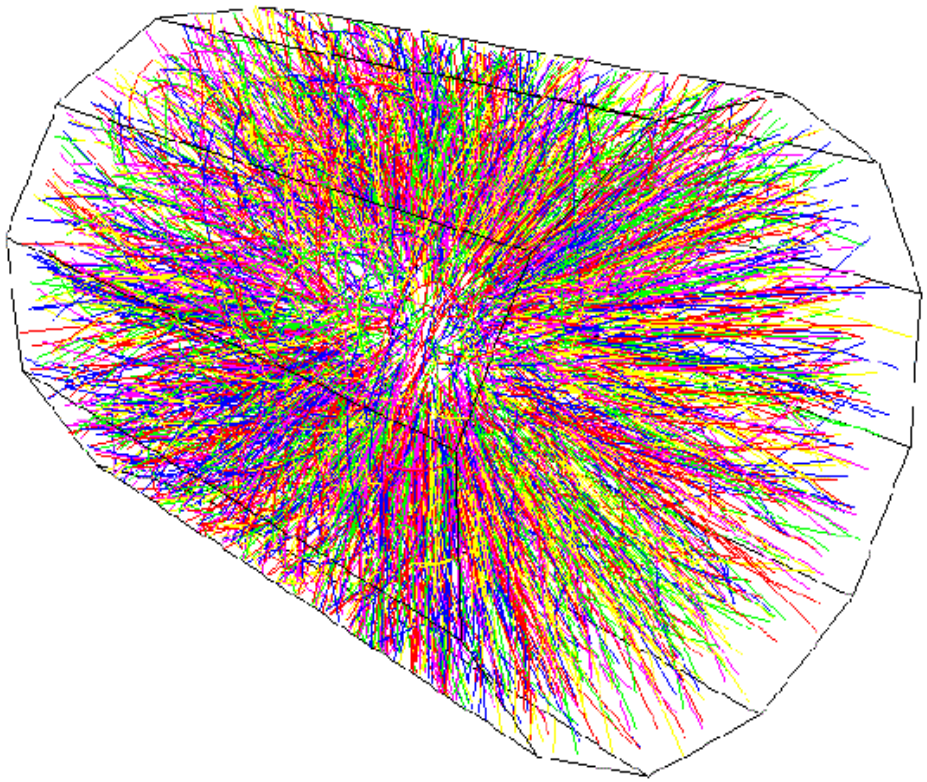
Collaboration	# members /institutions	Date of first data	# events/year	total data volume/year-TB
STAR	350/35	2000	10^7-10^8	300
PHENIX	350/35	2000	10^9	600
BABAR	300/30	1999	10^9	80
CLAS	200/40	1997	10^{10}	300
ATLAS	1200/140	2004	10^9	2000

STAR: Solenoidal Tracker At **RHIC**
RHIC: Relativistic Heavy Ion Collider

Physical Layout of the Detector



Result of Particle Collision (event)



Typical Scientific Exploration Process

- Generate large amounts of raw data
  - large simulations
  - collect from experiments
- Post-processing of data
  - analyze data (find particles produced, tracks)
  - generate summary data
    - e.g. momentum, no. of pions, transverse energy
    - Number of properties is large (50-100)
- Analyze data
  - use summary data as guide
  - extract subsets from the large dataset
    - Need to access events based on partial properties specification (range queries)
    - e.g. ((0.1 < AVpT < 0.2) ^ (10 < Np < 20)) v (N > 6000)
  - apply analysis code

Size of Data and Access Patterns

- STAR experiment
  - 10^8 events over 3 years
  - 1-10 MB per event: reconstructed data
  - events organized into 0.1 - 1 GB files
  - 10^{15} total size
  - 10^6 files, ~30,000 tapes (30 GB tapes)
- Access patterns
  - Subsets of events are selected by region in high-dimensional property space for analysis
  - 10,000 - 50,000 out of total of 10^8
  - Data is randomly scattered all over the tapes
- Goal: Optimize access from tape systems

EXAMPLE OF EVENT PROPERTY VALUES

I event 1
I N(1) 9965
I N(2) 1192
I N(3) 1704
I Npip(1) 2443
I Npip(2) 551
I Npip(3) 426
I Npim(1) 2480
I Npim(2) 541
I Npim(3) 382
I Nkp(1) 229
I Nkp(2) 30
I Nkp(3) 50
I Nkm(1) 209
I Nkm(2) 23
I Nkm(3) 32
I Np(1) 255
I Np(2) 34

I Np(3) 24
I Npbar(1) 94
I Npbar(2) 12
I Npbar(3) 24
I NSEC(1) 15607
I NSEC(2) 1342
I NSECpip(1) 638
I NSECpip(2) 191
I NSECpim(1) 728
I NSECpim(2) 206
I NSECkp(1) 3
I NSECkp(2) 0
I NSECkm(1) 0
I NSECkm(2) 0
I NSECp(1) 524
I NSECp(2) 244
I NSECpbar(1) 41
I NSECpbar(2) 8

R AVpT(1) 0.325951
R AVpT(2) 0.402098
R AVpTpip(1) 0.300771
R AVpTpip(2) 0.379093
R AVpTpim(1) 0.298997
R AVpTpim(2) 0.375859
R AVpTkp(1) 0.421875
R AVpTkp(2) 0.564385
R AVpTkm(1) 0.435554
R AVpTkm(2) 0.663398
R AVpTp(1) 0.651253
R AVpTp(2) 0.777526
R AVpTpbar(1) 0.399824
R AVpTpbar(2) 0.690237
I NHIGHpT(1) 205
I NHIGHpT(2) 7
I NHIGHpT(3) 1
I NHIGHpT(4) 0
I NHIGHpT(5) 0

54 Properties, as many as 10^8 events

Opportunities for optimization

- Prevent / eliminate unwanted queries
  => query estimation (fast estimation index)
- Read only events qualified for a query from a file (avoid reading irrelevant events)
  => exact index over all properties
- Share files brought into cache by multiple queries
  => look ahead for files needed and cache management
- Read files from same tape when possible
  => coordinating file access from tape

The Storage Access Coordination System (STACS)